Amendment to Specification:

Please amend the summary of invention beginning on page 6, line 1, as follows:

A system and method accepts communications directly into the addressable main memory of the computer system, bypassing the memory bus, while allowing retrieval at memory speeds. A pointer to the communication in memory is queued to allow processors to retrieve the communication without the use of interrupts. Each communication may be queued for a specific processor or other entity, such as the processor that handled the last communication from the same source and to the same destination that had the same protocol as the communication. When the processor or other entity processes the communication, it may provide a response or other communication into the addressable memory of the computer system. The system and method retrieves the response or other communication directly from the memory and provides it as output, for example, to a network. If multiple processors have access to shared memory and also have access to their own private memory, the first communication may be accepted into shared memory

and the response or other communication may be stored into private memory for the processor processing [he] the communication, spreading [he] the various communications across different memory buses. In the case in which the first communication is relatively small, such as a request for information, and the response or other communication is large, such as the information requested being a web page or a file from a storage device, processing all of the requests via a shared memory bus and the responses via private memory busses of multiple processors exploits the asymmetry of the communications, to provide a throughput larger than would be possible using a single bus without overwhelming any of the memory buses.

Please amend the paragraph beginning on page 21, line 3, as follows:

To complete the assignment of a communication to a processor **232-234**, processor A **230** places the pointer it copied from head/tail pointer storage **284** into processor B FIFO **286** or processor C FIFO [[**288**]] **287** depending on whether the assignment was made to processor B **232** or processor C **234**, respectively. The block itself may remain in buffer **280**, although in another embodiment, processor A

230 moves the block itself into Processor B FIFO **286** or processor C FIFO [[**288**]] **287** instead of the pointer or moves the block to a different buffer (not shown) that may be shared or dedicated to a processor and inserts into FIFO **286** or [[**288**]] **287** a pointer to the block in this buffer. In the embodiment in which a buffer is not dedicated to the processor, processor B FIFO **286** and processor C FIFO [[**288**]] **287** are arranged as conventional double buffers with a head and a tail pointer (in a reserved location of each respective FIFO **286**, [[**288**]] **287** not used to write pointers or blocks as shown in t he Figure) pointing to the head and tail of the FIFO and processor A **230** moves either the appropriate pointer or the block to the tail of the FIFO **286**, [[**288**]] **287** and updates the tail to the next available location in the FIFO. If a buffer is dedicated to each processor is used, head and tail pointers are used to point to the head and tail of the buffer itself, with the buffer acting as the FIFO in a double buffer arrangement as described above.

Please amend the paragraph beginning on page 21, line 4 as follows:

In the embodiment in which processor A **230** assigns the block to the processor having the shortest queue, processor A **230** may determine which queue is shortest by comparing the difference between the head and the tail of each FIFO **286**, [[**288**]] **287** and choosing the processor **232-234** corresponding to the FIFO having the smallest difference.

Please amend the paragraph beginning on page 22, line 17 as follows:

Processors **232-234** monitor their respective FIFOs and when the head and tail are not pointing to the same position, retrieve either the block at the head of the FIFO **286**, [[**288**]] **287** or the block in buffer **280** pointed to by the head of the respective FIFO **286**, [[**288**]] **287**, (or in another FIFO as described above) and then advance the head of the respective FIFO **286**, [[**288**]] **287** past the end of the pointer or block stored in the respective FIFO **286**, [[**288**]] **287**. In another embodiment, each processor **232**, **234** reads from a fixed location of shared memory interface **218** and receives any block corresponding to the pointer at the head of the FIFO **286**, [[**288**]] **287** corresponding to the location from which the block was read. If the block is not empty, processor **232**, **234** processes the block as described below.

In such embodiment, shared memory interface **218** supplies the block from the buffer **280** and manages the head and tail of the FIFOs **286, 288.**

Please amend the paragraph beginning on page 24, line 1 as follows:

In one embodiment, incoming dispatch storage **214** has the arrangement shown in Figure **2D.** In such embodiment, instead of head/tail pointer storage **284,** available locations FIFO **292** is employed. Buffer **280** need not be arranged as a circular buffer. Instead, buffer **280** contains fixed-length blocks, and pointers to each such block are initialized into available locations FIFO **292,** which is arranged as a circular buffer with pointers to the head and tail of the FIFO **292.** (All initializations of incoming dispatch storage **214** may be performed by incoming interface manager **212** on power on or reset.) When incoming interface manager **212** receives a communication, it extracts the portion to be saved as a block and retrieves from available locations FIFO **292** the pointer to buffer **280** pointed to by the head of available locations FIFO **292,** stores the block at the location corresponding to the pointer removed, and advances the head of available

locations FIFO **292** to point to the next location in available locations FIFO **292**. If incoming interface manager **218** routes the block as described above, the pointer removed is placed at the location of processor B FIFO **286** or processor C FIFO [[**288**]] **287** indicated by the respective FIFO's tail corresponding to the processor **232**, **234** to which the block was assigned.

Please amend the paragraph beginning on page 25, line 1 as follows:

If one of the processors **230-234** assigns the block to a processor, an additional FIFO (not shown) may be used that is arranged as FIFOs **286** and [[**288**]] **287** described above, and incoming interface manager **212** places the pointer it retrieves from available locations FIFO **292** onto the additional FIFO at the location indicated by its tail and advances the tail. Processor A **230** assigns blocks by retrieving them from the head of the additional FIFO, advancing its head, and then adding the pointer to the tail of the FIFO **286**, [[**288**]] **287** corresponding to the processor to which the block was assigned as described above.

Please amend the paragraph beginning on page 25, line 12 as follows:

When a processor **232, 234** retrieves for processing the block pointed to by the pointer to the buffer **280** at the head of its respective FIFO **286,** [[**288**]] **287** it places the pointer at the location pointed to by the tail of available locations FIFO **292** and advances the tail to the next location in the FIFO, signaling that the block in buffer **280** may now be reused.

Please amend the paragraph on page 25, line 20 as follows

To generate output to an external system, processor B **232** and processor C **234** write any response to the block to their respective private memory interface **250** or **260,** each providing a conventional memory interface to the private memory bus of processor B **232** and processor C **234,** respectively, of front side bus **220.** Private memory interfaces **250, 260** each pass the response received to a location in a buffer at the tail of outgoing storage B and C **252, 262,** respectively. Each outgoing storage **252, 262** is a conventional storage device, such as memory or disk storage, arranged as a conventional double buffer, with a

head and a tail as described above with respect to FIFOs **286**, [[**288**]] **287**. As will be described below, it is not necessary to utilize separate outgoing storages **252**, **262** as a single outgoing storage can be used, with each private memory interface **250**, **260** writing into the single outgoing storage **252** or **262**, although it can simplify the circuitry to have individual outgoing storages **252**, **262** to avoid contention issues. The responses are written in blocks having a fixed or variable length as described above. In one embodiment, each processor **234**, **236** writes to a single memory location, and the respective private memory interface **250**, **260** manages placing it into a buffer in the respective outgoing storage **252**, **262** and updating the respective tail pointer. In another embodiment, each processor **232**, **234** reads the tail in the respective outgoing storage **252**, **262** and stores the response or other communication into the private memory buffer location in the respective outgoing storage **252**, **262** indicated by the tail pointer in the respective outgoing storage **252**, **262**, and updates the tail pointer.

Please amend the paragraph beginning on page 27, line 3 as follows:

Outgoing storages **252, 262** are monitored by one or more outgoing interface manager **254, 264**. Each outgoing interface manager **254, 264** checks the head and tail pointer for the outgoing storage **252, 262** it is monitoring, and if they are different, it takes the block indicated by the tail pointer in outgoing storage **252, 262** in a manner similar to that described above for FIFOs **286,** [[**288**]] **287,** provides the block to an outgoing communication interface **256** or **266** described in more detail below, and updates the tail pointer in the respective outgoing storage **252, 262** to point to the next block in that outgoing storage **252, 262**.